LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-PROC-208287

WxWindows Interface for CALE

Paul Amala, Christopher Egner, Jeffery Hagelberg

December 1, 2004

Nuclear Explosives Code Development Conference (NECDC 2004)
Livermore, CA, United States
October 4, 2004 through October 8, 2004

This work was performed under the auspices of the U. S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

WxWindows Interface for CALE (U)
UCRL-XXXXX

Paul Amala*, Christopher Egner†, Jeffery Hagelberg††

*Lawrence Livermore National Laboratory; † Rochester Institute of Technology, Rochester NY; †† University of California - Davis

wxWindows is an Open Source, platform independent, User Interface (UI) which has been in development for over eleven years (http://www.wxwindows.org). Currently wxWindows is actively supported for the Linux/Unix (X11, Motif and GTK+), Mac OS 9 and X, all Win32 OSes, MGL, and OS/2 operating systems. wxWindows is written in C++ using an object oriented programming framework; it is a reasonably lightweight API (called wxWidgets) sitting over the native graphics packages of the various platforms it supports.

The original version of CALE was written for the basic target platform of Unix using X11 as the graphics package. There have been separate efforts to port the code to Mac OS 9, Mac OS X, Win32, Windows Services for Unix (SFU) and CygWin. Each of these used a variety of different graphical interface approaches and build/make systems. For instance Windows SFU and CygWin could still only use X11 graphics. So could the Win32 version, if a X11 server library and client software were installed. A native Win32 version of CALE was contemplated, but never started. The Macintosh versions were completed but never widely distributed to the users. Given the growing code version support issues, and the slow deviation from the portable code model CALE originally started with, it was desired to come up with a simple graphical UI that would be cross platform portable with only a single code base and build system.

During the past two summers, two Laboratory summer students and a CALE team code physicist have worked on porting CALE to the wxWidgets UI. In the summer of 2003 Jeffery Hagelberg (formerly Purdue University, now at the University of California-Davis) started the project. During the spring & summer of 2004 Christopher Egner (Rochester Institute of Technology) completed the work. Paul Amala (A/X-Program at LLNL) supervised the students for their combined 30 weeks of effort.

This poster session describes the wxWindows interface as it is implemented in CALE, the level of cross platform portable it actually affords, and the lessons learned during the porting of an existing X11 program to this open source software package. (U)

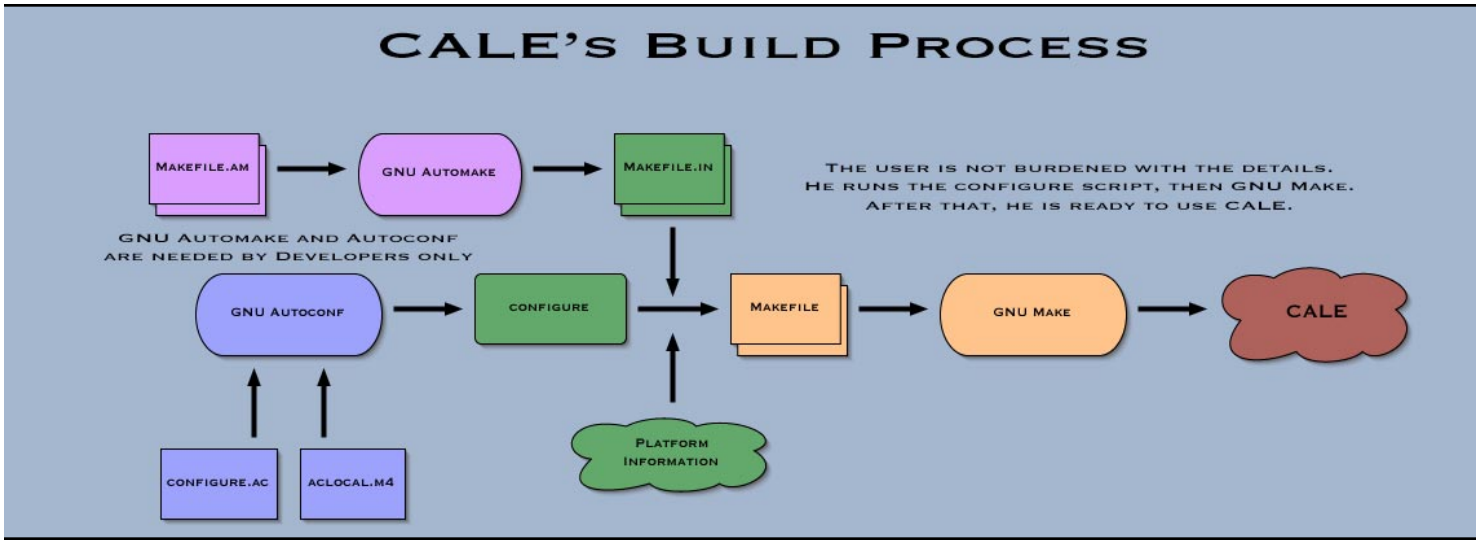
Introduction

This project began several years ago when it was desired to come up with a common code base for CALE, which was portable to all platforms that CALE might run upon. As the three physicists who work upon CALE were unable (due to time and interest constraints) to make this effort, it was decided that this relatively computer science intensive project would be a good summer job for some laboratory summer students.

During the first summer of 2003, Jeffery Hagelberg was hired to do this work. Moving from the X11 base of the code to a platform independent version was *significantly* more work than anticipated. By the end of his 10-week summer term he had a working version running on the Win32 platform with a limited functional implementation of the graphical CALE interface.

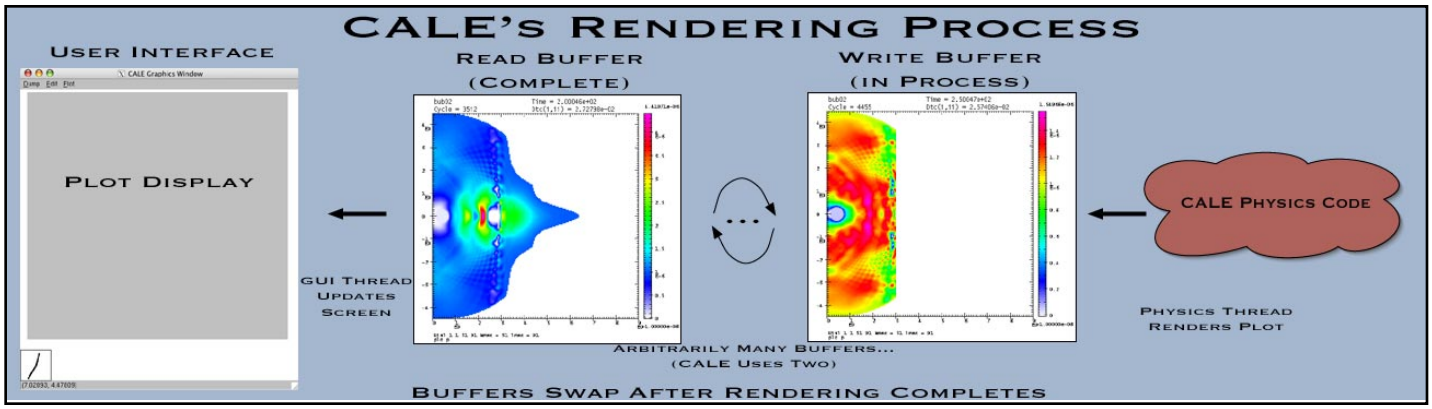
The second iteration on this work was the following spring and summer of 2004 when an intern was hired for 20 weeks to complete the project. Chris Egner updated the project to the latest version of wxWidgets and completed the porting to the Win32, Linux, Tru_64, and Mac OS X platforms. In doing this work, he also completely overhauled the build system for CALE, moving it from a customized m4 configuration to an automated GNU autotools setup (see figure below). It was found that this substantially more general build system was necessary for building upon the various platforms while still giving the flexibility of using both the wxWidgets and X11 API's.

CALE's new build process



Lessons Learned

- Buffered graphical I/O is a must
- Threaded implementation is also important
- Raw X11 is still faster than a universal API, but not as portable
- C language discipline is very important when using C++ compilers
- WxWidgets is still immature after 11 years



The first thing that was found when using the wxWidget API was that the graphics were unacceptably slow. It was determined that a buffered I/O system would be much more efficient. The figure above illustrates the method used. In the original X11 implementation this was never an issue; with added overhead of wxWidget's C++ API this became a critical speed bottleneck.

Basically, while a new frame is being rendered in memory, the current frame is being displayed. When an update is needed, the frame in memory is then displayed and the memory is recovered for future use. This model is commonly used with the native Win32 graphical API's, among others.

The second item is that originally CALE was written to be a console-based application that called X11 as a library to display graphics only. However, like many modern 'event based' API's, wxWidgets wants to be in the driver's seat too. This led the design to of the CALE/wxWidgets based version to be thread based. The base CALE physics code with its associated routines runs in one thread, the wxWidgets API routines run in a second thread, and the graphics rendering runs in a third thread. Unfortunately this model is somewhat problematic in the current implementation of wxWidgets on Mac OS X.

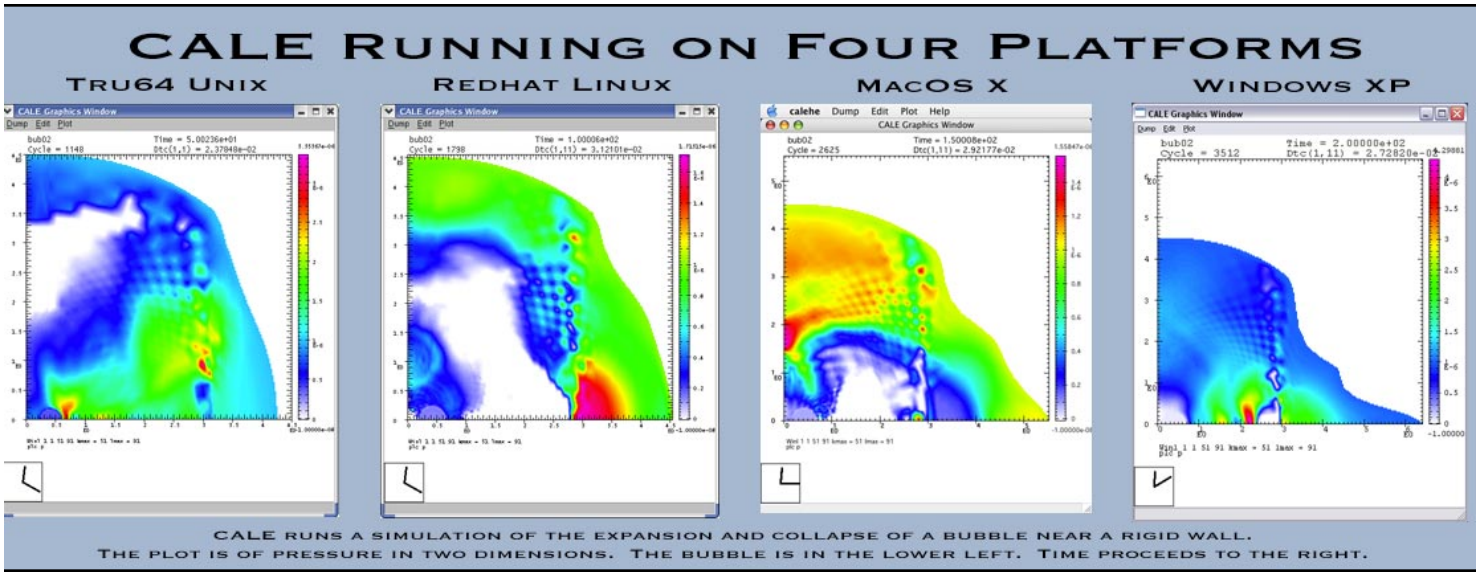
The third lesson is that we could not get the wxWidgets implementation to run as fast as the raw X11 implementation had. This was even true with 'backend' implementations such as Windows Services for Unix (Unix kernel running in parallel with a Win NT kernel, using a separate X11 server/client such as X-Win32) or CygWin (Unix core ported to Win32 platform). While it wasn't tested, now that current versions of Mac OS X come with X11 libraries, the same concerns are present; a direct X11 port to Mac OS X probably will run faster. In our testing with Windows, Linux and Tru_64 (i.e. Dec Alpha) platforms it was consistently found that wxWidgets was approximated 25-33% slower in graphical performance. Of course physics performance was unaffected. The slower graphical performance was disappointing, given that interactive graphics was the main envisioned use of the common API interface. It is yet to be decided if this slow down is acceptable, given that the common single code base and build system is still a big win on the multiple desired platforms supported.

The fourth lesson learned was somewhat interesting. A lot of effort had been made over the past several years to make CALE ISO C compliant. However, when we began using the C++ compilers (gcc, Intel C++, Compaq cc, etc.) on the base code, all kinds of nits began to show up. It was not appreciated that to make CALE's base C code compatible with wxWidget's C++ API that CALE would have to be more meticulously made to conform to the modern C standards. While this was unexpected it was not a wasted effort. No matter what the end conclusion in using the wxWidgets API, we want to make CALE as portable and code safe as possible; finding subtle problems with our C code is desirable. It is possible that a more mature, professional API is a better way to go (see comments below).

The most disappointing conclusion to the over-all project however is how immature we found wxWidget to be. The documentation is very spotty, and complete sections are missing. Much of the actual documentation still exists only the comments within the code itself. On-line news groups and mailing lists were a help, but not equal to well written documentation. Also the API is unevenly implemented. Substantial parts of the Mac OS X interface are not done. The base X11 calls are being deprecated in favor of the GTK+ ones. The latest version of wxWidgets is fundamentally broke in various parts; when Chris Egner was done for the summer he warned us that we would need to update to the newer version when available and that this would probably entail more rewriting of the glue code linking the CALE physics routines to the graphical routines.

The most unfortunate part of all of this is that by looking at the documentation and the base source code one cannot always see this – a call that may work perfectly well under Win32 will crash a run under Mac OS X. This of course does not encourage one attempting to make a single platform independent code base. And to figure out what is actually wrong takes a combination of looking through the imcomplete documentation, looking at the source code, and asking questions on news groups on-line.

However, on the positive side, the API does give a consistent look and feel on the various platforms:



Conclusions

When this project was first considered in the early spring of 2003, QT3 was not available yet for Mac OS X. Since this was one of our base platforms this was a showstopper. Now in the late fall of 2004 QT3 is available for all the platforms of interest. QT3's main disadvantage is its moderately expensive licensing fee if one does not want to distribute their source code under an open source type license (which we do not want). Its great advantages however are that it is well documented, it is a commercial product with professional technical support to answer questions without having to avail oneself of a hit or miss mailing list, and the availability of high quality text books to teach oneself how to use the API. All of these facets were lacking in our use of wxWidgets.

This is of course quite a bit of "Monday Morning Quarterbacking", stating that a totally different API could solve many of our problems with wxWidgets. Thirty weeks of effort went into getting an almost fully functional platform independent version of CALE up and running. But the unfortunate fact of the matter is that quite a bit of that effort was taken up in dealing with bugs in the wxWidgets API, spotty documentation, and an uneven level of implementation that left one feeling that somehow the developers missed the 'platform independence' part of their claims of platform independence. Caveat emptor.

Student Follow Up

Jeffery Hagelberg is now working with Andrew Cook (AX-Div/LLNL) on the Miranda project, attempting to improve parallel I/O performance with the PACT libraries. He is working on his Master's Degree through the UC-Davis Dept. of Applied

Science. Chris Egner, upon graduating from R.I.T., has accepted a flex position at LANL working with the Biometrics group on Natural Language Programming. He hopes to continue with his graduate education in the fall of 2005 in this field.

Acknowledgements

The authors would like to thank Lila Chase, Robert Managan and Robert Tipton for their help during this work.

References

WxWidgets: http://www.wxwindows.org

QT3: http://www.trolltech.com

Windows Services for Unix: http://www.microsoft.com/windows/sfu/

CygWin: http://www.cygwin.com/

X-Win32: http://www.starnet.com